Tutorial: Windows Mobile Application Development using Custom Development

Sybase Unwired Platform 1.5.5

DOCUMENT ID: DC01285-01-0155-02
LAST REVISED: February 2011

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

Introduction to Getting Started Tutorials

Getting started tutorials enable users of all levels to try Sybase® Unwired Platform with minimal setup. You can also use the tutorials to demonstrate system functionality and train users.

Overview of Getting Started Tutorials

The getting started tutorials demonstrate how to develop, deploy, and test mobile business objects, device applications, and message-based mobile workflow packages.

- Learn mobile business object (MBO) basics, and create a mobile device application:
  - *Tutorial: Mobile Business Object Development*
  - *Tutorial: BlackBerry Application Development using Device Application Designer*
  - *Tutorial: Windows Mobile Device Application Development using Device Application Designer*
- Create native mobile device applications:
  - *Tutorial: BlackBerry Application Development using Custom Development*
  - *Tutorial: iPhone Application Development using Custom Development*
  - *Tutorial: Windows Mobile Application Development using Custom Development*
- Create a mobile workflow package:
  - *Tutorial: Mobile Workflow Package Development*

The getting started tutorials demonstrate a cross section of basic functionality, which includes creating MBOs that can be used in replication-based or message-based synchronization; and using various Sybase Unwired WorkSpace development tools, independent development environments, and device types.

Table 1. Tutorial summary

Tutorials	Mobile business objects (MBOs)	Synchronization types	Development tools	Device types
Tutorial: Mobile Business Object Development	Create new MBOs	Replication-based	Sybase Unwired WorkSpace	N/A
Tutorial: BlackBerry Application Development using Device Application Designer	Reuse MBOs	Replication-based	Device Application Designer	BlackBerry

Tutorials	Mobile business objects (MBOs)	Synchronization types	Development tools	Device types
Tutorial: BlackBerry Application Development using Custom Development	Create new MBOs	Replication-based	Sybase Unwired WorkSpace	BlackBerry
Tutorial: iPhone Application Development using Custom Development	Create new MBOs	Message-based	Sybase Unwired WorkSpace	iPhone
Tutorial: Windows Mobile Application Development using Device Application Designer	Reuse MBOs	Replication-based	Device Application Designer	Windows Mobile
Tutorial: Windows Mobile Device Application Development using Custom Development	Create new MBOs	Message-based	Sybase Unwired WorkSpace	Windows Mobile
Tutorial: Mobile Workflow Package Development	Create new MBOs	Message-based	Mobile Workflow Forms Editor	Windows Mobile

Understanding the Unwired Platform Development Environment

Learn more from the getting started tutorials by understanding basic development environment concepts. Sybase Unwired Platform provides an Eclipse development environment.

Development in Eclipse

Sybase Unwired WorkSpace is a plug-in to your Eclipse development environment that provides tools for creating mobile applications.

Unwired WorkSpace includes back-end integration tools that connect Unwired Server to enterprise data sources, allowing you to create mobile business objects (MBOs) from the back-end business data model.

Developers can perform MBO code generation at any time and use this MBO model code along with the user interface code that users write in a native integrated development environment (IDE). This makes the code available to transition from the Unwired WorkSpace MBO development tool to the fully extensible and open development environment provided for device platforms from third-party vendors. Optionally, use the Device Application

Designer to create prototype device applications for BlackBerry and Windows Mobile devices.

Developers can use the Mobile Workflow Forms Editor to develop message-based mobile workflow packages for Windows Mobile and iOS devices.

Understanding Fundamental Mobile Development Concepts

Learn more from the getting started tutorials by understanding basic mobile development concepts.

Learn more about these concepts:

- *Fundamentals*
- *Sybase Unwired WorkSpace – Mobile Business Object Development*

Mobile Business Objects

Mobile business objects help form the business logic for mobile applications.

A mobile business object (MBO) is derived from a data source (such as a database server, Web service, or SAP® server). MBOs are deployed to Unwired Server, and accessed from mobile device application clients. MBOs include:

- Implementation-level details – metadata columns that include information about the data from a data source.
- Abstract-level details – attributes that correspond to instance-level properties of a programmable object in the mobile client, and map to data source output columns. Parameters correspond to synchronization parameters on the mobile client, and map to data source arguments. For example, output of a SQL SELECT query are mapped as attributes, and the arguments in the WHERE clause are mapped as synchronization parameters, so that the client can pass input to the query.

  MBO operations include parameters that map to data source input arguments. Operation parameters determine information a client passes to the enterprise information system (EIS).
- Relationships – defined between MBOs by linking attributes and parameters in one MBO, to attributes and parameters in another MBO.

You can define MBOs using either a top-down approach—first designing attributes and parameters, then binding them to a data source; or a bottom-up approach—first specifying a data source, then automatically generating attributes and parameters from it.

A mobile application package includes MBOs, roles, and data source connection mappings, and other artifacts that are delivered to the Unwired Server during package deployment.

Synchronization Methods

Developers can use either replication-based or message-based synchronization to move data and transactions between device application clients and Unwired Server.

The choice depends on the target device platform, application requirements, target platform, and the nature of data changes and activity between Unwired Server and clients, for example, mobile workflow forms always use message-based synchronization.

Unwired Server manages and maintains data freshness between multiple data sources and device application clients through synchronization.

Application Types

Sybase Unwired Platform supports two choices for application type. First is the native application type, and the other is the container-based business workflow type.

The native application model enables the developer to write custom code (C#, Java, or Objective-C, depending on the platform) to create a device application. The native application model is supported on BlackBerry, iOS, Windows Mobile, and Windows platforms. The choice depends on the functionality desired in the application, and the need to access third-party and platform-provided APIs. Optionally, use the Device Application Designer to create prototype device applications for BlackBerry and Windows Mobile devices.

The business workflow model offers a fast and simple way to build applications that support simple business workflows, such as approvals and requests. The workflow model is supported on iOS, Windows Mobile, and Windows platforms.

Data Sources

A data source is the enterprise information system where data is retrieved from and transactions are executed. A connection profile is a design-time connection to a data source. Connection profiles are created to specific data source by providing connection information such as host, port, login, and password among others. The connection profiles are used to define MBOs and operations, and mapped to existing, or used to create new, server connections when the package is deployed to Unwired Server..

Unwired Platform hides the interaction complexity with datasource-specific protocols, such as JDBC™ for database and SOAP for Web services.

Unwired Platform currently supports multiple EIS connection types. See *Supported Third-Party Software and Hardware* for information.

Switching Between Developer Profiles

Switch between basic and advanced developer profiles in the Mobile Application Diagram.

If you do not see an Unwired WorkSpace feature (wizard, property, or WorkSpace Navigator item) that you expect or need, switch to the advanced developer profile, or modify developer profile settings. To use backend data sources other than those supplied by Sybase Unwired Platform, you must switch to the advanced developer profile to see the Server Connection Mapping page when deploying the Mobile Business Object package.

1. Right-click in the Mobile Application Diagram and select **Switch Developer Profile > Basic/Advanced**.
2. You can also select **Window > Preferences > Sybase, Inc. > Mobile Development > Developer Profile** to directly view or modify the developer profile preference settings. Basic is the default developer profile.

Task Flow

Sybase Unwired WorkSpace Eclipse tutorials explain how to develop, deploy, and run a mobile application.

Table 2. Eclipse tutorials

Task	Goals	Steps required to complete the task
Getting Started	• Install all required WorkSpace components and external resources. • Start Unwired Server, then use Sybase Control Center to connect to the server. • Open the Mobile Development perspective, and become familiar with the views of the perspective, the Mobile Application Diagram, and the Device Application Designer.	• *Installing Sybase Unwired Platform* on page 9 • *Installing Microsoft ActiveSync* on page 9 • *Starting Unwired Platform Services* on page 10 • *Starting Sybase Unwired WorkSpace* on page 10 • *Connecting to Sybase Control Center* on page 11 • *Learning the Basics* on page 11 **Note:** These steps are prerequisites for the rest of this tutorial. You need to perform them only once.
Developing Database Mobile Business Objects	• Create a mobile application project and a connection to the database. • Create two mobile business objects, and create a relationship between them. • Deploy the mobile business objects to Unwired Server.	Complete the *Tutorial: Mobile Business Object Development*.

Task	Goals	Steps required to complete the task
Developing a Windows Mobile Device Application	• Generate code for the Windows Mobile platform, code a Windows Mobile device application, and run it on an emulator.	• *Deploying the Database Mobile Business Objects* on page 15 • *Generating C# Object API Code* on page 17 • *Creating the User Interface for the Windows Mobile Device Application* on page 19 • *Installing Sybase Messaging Runtime* on page 30 • *Registering the Emulator in Sybase Control Center* on page 30 • *Configuring Connection Settings* on page 32 • *Deploying and Running the Device Application* on page 32

Getting Started

Goal: Install and learn about Sybase Unwired Platform and its associated components.

The following tasks are required, unless otherwise noted, for all tutorials, but you need to perform them only once.

1. *Installing Sybase Unwired Platform* on page 9
2. *Starting Unwired Platform Services* on page 10
3. *Starting Sybase Unwired WorkSpace* on page 10
4. (optional) *Learning the Basics* on page 11

Installing Sybase Unwired Platform

Goal: Install Sybase Unwired Platform.

Install these Sybase Unwired Platform components:

- Data Tier
- Unwired Server
- Unwired WorkSpace
- Device Application Designer
- Windows Mobile .NET components (for developing device applications in Visual Studio)

If Unwired Platform is already installed and any of these components are missing:

1. Start the Sybase Unwired Platform installer.
2. Follow the instructions in the installation wizard.
3. Select the required components, and complete the installation.

For complete installation instructions, see the *Sybase Unwired Platform Installation Guide* and *Release Bulletin*.

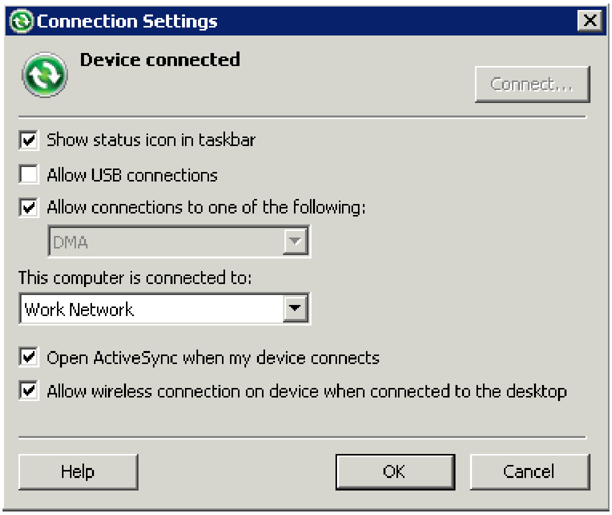
Installing Microsoft ActiveSync

Goal: Install and configure Microsoft ActiveSync so you can deploy and run a mobile application on a Windows Mobile emulator.

Note: Microsoft ActiveSync is for Windows XP. If you are using Windows Vista, Windows 7, or Windows 2008, install Virtual PC 2007 SP1 and Windows Mobile Device Center to manage synchronization settings. Download the Windows Mobile Device Center from *http://www.microsoft.com/windowsmobile/en-us/downloads/microsoft/device-center-download.mspx*.

1. Download Microsoft ActiveSync from *http://www.microsoft.com/windowsmobile/en-us/help/synchronize/device-synch.mspx*. Save it to your local machine. Windows XP requires ActiveSync version 4.5.
2. In Windows Explorer, double-click **setup.msi**.
3. When installation is complete, restart your machine.

   ActiveSync starts automatically, and its icon appears in the Windows toolbar.
4. Double-click the **ActiveSync** icon.
5. Select **File** > **Connection Settings**.
6. Select **Allow Connections to One of the Following**, then select **DMA**.
7. Under This Computer is Connected to, select **Work Network**.

   Device connected

   ☑ Show status icon in taskbar

   □ Allow USB connections

   ☑ Allow connections to one of the following:

   DMA

   This computer is connected to:

   Work Network

   ☑ Open ActiveSync when my device connects

   ☑ Allow wireless connection on device when connected to the desktop

8. Click **OK**.

Starting Unwired Platform Services

Goal: Start Unwired Server and the sample database.

In Windows, select **Start** > **Programs** > **Sybase** > **Unwired Platform<version>** > **Start Unwired Platform Services** .

Starting Sybase Unwired WorkSpace

Goal: Start Unwired WorkSpace.

1. In Windows, select **Start** > **Programs** > **Sybase** > **Unwired Platform<version>** > **Unwired WorkSpace**.

Sybase Unwired WorkSpace opens, and displays the Welcome page with links to product information, and to the product.

2. To read more about Sybase Unwired WorkSpace concepts and tasks, select **Help > Help Contents** from the main menu.

Connecting to Sybase Control Center

Goal: Open the Web-based Sybase Control Center administration console to manage Unwired Server and its components.

From Sybase Control Center, you can:

- View servers and their status
- Start and stop a server
- View server logs
- Deploy a mobile application package
- Set role mappings

1. Select **Start > Programs > Sybase > Sybase Control Center**.

   Note: If Sybase Control Center does not launch, make sure that the Sybase Unified Agent service is started. See the Installation Guide for details.

2. Log in using the default login:
   - User Name – `supAdmin`
   - Password – `s3pAdmin`

   Logging in to Sybase Control Center (SCC) allows you access to Unwired Platform administration features that you have been authorized to use. Administrators of any Sybase product can log into SCC. However, only users assigned to the Super Administrator or Domain Administrator roles for Unwired Platform can log in to Unwired Server from Sybase Control Center.

   Logging in to SCC only allows you access to the SCC interface. If Unwired Server has not been authenticated, you will not be able to see or administer any resources.

3. Select **Help > Online Documentation** for additional information on configuring, managing, and monitoring Unwired Server.

Learning the Basics

Goal: Learn about Sybase Unwired WorkSpace and how to access help.

Prerequisites

Start Unwired WorkSpace.

Task

1. From the Welcome page, select any of the links to familiarize yourself with the Unwired WorkSpace environment.

   To close this page, click the **X**. You can reopen this page by selecting **Help > Welcome**.
2. Select **Start Development** to access the Sybase Unwired WorkSpace development environment. Look at the area (window or view) that you will be working in to access, create, define, and update mobile business objects (MBOs).

View	Description
WorkSpace Navigator	This view displays mobile application project folders, each of which contains all project-related resources in subfolders, including MBOs, data source references to which the MBOs are bound, personalization keys, and so on. Use this view to review and modify MBO-related properties.
Enterprise Explorer	A window that provides functionality to connect to various enterprise back-end systems; for example, database servers, SAP servers, and Sybase Unwired Server.

View	Description
Mobile Application Diagram	A graphical editor where you create and define mobile business objects. Use the Mobile Application Diagram to create MBOs (including attributes and operations), then define relationships with other MBOs. You can: • Create MBOs in the Mobile Application Diagram using Palette icons and menu selections – either bind or defer binding to a data source, when creating an MBO. For example, you may want to model your MBOs before creating the data sources to which they bind. This is sometimes called the top-down approach. • Drag items from Enterprise Explorer and drop them onto the Mobile Application Diagram to create the MBO – quickly creates the operations and attributes automatically based on the data source being dropped on the Mobile Application Diagram. This is sometimes called the bottom-up approach. Each new mobile application project generates an associated Mobile Application Diagram.
Palette	Access the Palette from the Mobile Application Diagram. It provides controls, such as the ability to create MBOs, add attributes and operations, and define relationships, by dragging and dropping the corresponding icon onto the Mobile Application Diagram or existing MBO.
Properties view	Select an object in the Mobile Application Diagram to display and edit its properties in the Properties view. You cannot create an MBO from the Properties view, but generally, most development and configuration is performed here.
Outline view	Displays an outline of the file that is currently open in the editor area, and lists structural elements. The contents are editor-specific.
Problem view	Displays problems, errors, or warnings that you may encounter.

3. To access the online help, select **Help** > **Help Contents** from the main menu bar.

4. Expand any of the documents that appear in the left pane. Some documents are for Sybase Unwired Platform, while others are for the Eclipse development environment.

Developing a Windows Mobile Device Application

Generate code for the Windows Mobile platform, develop a Windows Mobile device application with code, and test its functionality.

Prerequisites

Complete these tasks:

- *Getting Started* on page 9
- Finish the *Tutorial: Developing Database Mobile Business Objects*.
- Install Windows Mobile 6 Professional SDK.
- Install Microsoft .NET Compact Framework 3.5
- The Sybase Unwired Platform Windows Mobile .NET components must be installed. See *Installing Sybase Unwired Platform* on page 9.
- Visual Studio 2008 must be installed.

Task

Complete these tasks to develop the device application:

1. Open the SUP101 Mobile Application Project if it is not already open:

   In WorkSpace Navigator, right-click the **SUP101** folder and select **Open in Diagram Editor**.
2. *Deploying the Database Mobile Business Objects* on page 15
3. *Generating C# Object API Code* on page 17
4. *Creating the User Interface for the Windows Mobile Device Application* on page 19
5. *Installing Sybase Messaging Runtime* on page 30
6. *Registering the Emulator in Sybase Control Center* on page 30
7. *Configuring Connection Settings* on page 32
8. *Deploying and Running the Device Application* on page 32

Deploying the Database Mobile Business Objects

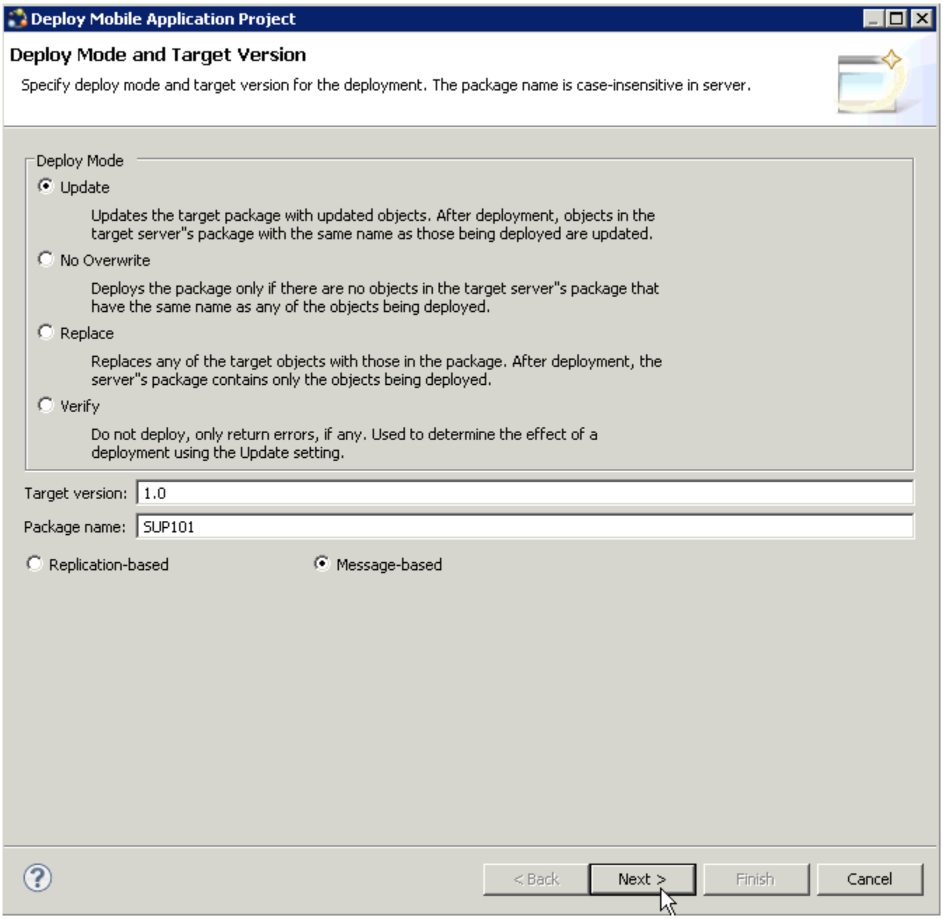
Goal: Deploy the project that contains the database mobile business objects to the server.

Prerequisites

Finish the *Tutorial: Developing Database Mobile Business Objects*. You must be connected to both the sampledb database and Unwired Server.

Task

1. Right-click in the SUP101 Mobile Application Diagram , and select **Deploy Project**.
2. On the first page of the Deploy Mobile Application Project, accept the defaults, select **Message-based**, and click **Next**.



Deploy Mobile Application Project

Deploy Mode and Target Version

Specify deploy mode and target version for the deployment. The package name is case-insensitive in server.

Deploy Mode

● Update

Updates the target package with updated objects. After deployment, objects in the target server"s package with the same name as those being deployed are updated.

○ No Overwrite

Deploys the package only if there are no objects in the target server"s package that have the same name as any of the objects being deployed.

○ Replace

Replaces any of the target objects with those in the package. After deployment, the server"s package contains only the objects being deployed.

○ Verify

Do not deploy, only return errors, if any. Used to determine the effect of a deployment using the Update setting.

Target version: 1.0

Package name: SUP101

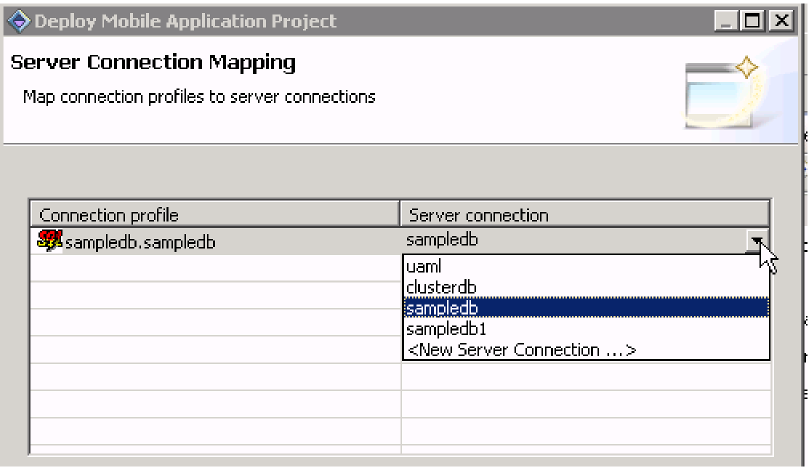
○ Replication-based ● Message-based

3. On the Contents page, select the **customer** and **sales_order** MBOs and click **Next**.
4. On the Package Jars page, click **Next**.

   Note: This window appears only if you are using the Advanced developer profile.

5. On the Target Server page, from the list of available servers, select **My Unwired Server** and click **Refresh** (or connect to the server, if not already connected).

   Once connected, accept the default Domain and Security configuration settings, and click **Next**.
6. If you have multiple server connections, you see the Server Connection Mapping page. Select the **sampledb** server connection and click **Finish**.



Deploy Mobile Application Project

Server Connection Mapping

Map connection profiles to server connections

Connection profile	Server connection
sampledb.sampledb	sampledb

7. When the Deployment status window shows the deployment was successful, click **OK**.
8. Connect to Unwired Server and view the deployed project.
   a) In the Enterprise Explorer, click **My Unwired Server**.

      My Unwired Server is a default Unwired Server connection profile that provides access to Unwired Server, which you started in a previous step.
   b) Expand **Domains** > **default** > **Packages**. The server package *sup101:1.0*, into which you deployed the MBOs, appears in the Packages folder. The two MBOs appear in the `Mobile Business Objects` folder.

Generating C# Object API Code

Generate Object API code for Windows Mobile devices.

Prerequisites

You must be connected to both the sampledb database and Unwired Server.

Task

1. In WorkSpace Navigator, right-click the **SUP101** project folder and select **Generate Code** to launch the Code Generation wizard.
2. Select **Continue without a configuration** and click **Next**.

   Note: This page of the code generation wizard is seen only if you are using the Advanced developer profile.
3. In Select Mobile Objects, select the **customer** and **sales_order** MBOs in the SUP101 mobile application project.

Dependent MBOs are automatically added (or removed) from the Dependencies section depending on your selections.

Note: Code generation fails if the server-side (run-time) enterprise information system (EIS) data sources referenced by the MBOs in the project are not running and available to connect to when you generate object API code.

4. Click **Next**.
5. Enter the information for these configuration options:

Option	Description
Language	Select **C#**.
Platform	Select **NET Compact Framework 3.5 for Windows Mobile**.
Unwired Server	Select the **My Unwired Server** connection profile.
Server domain	Select **default**.
Page size	Leave blank. This will be a message-based application, and the page size option is not enabled for message-based applications.
Namespace	Enter `SUP101`.
Destination	Specify the destination of the generated device client files, `\SUP101`. Select **Clean up destination before code generation** to clean up the destination folder before generating the device client files.
Replication-based	Unselect this option. This tutorial uses the **Message-based** synchronization option.
Message-based	Select this option to use message-based synchronization.
Backward compatible	This option is disabled.

6. Select **Generate metadata** to generate metadata for the attributes and operations of each generated client object.
7. Click **Finish**.

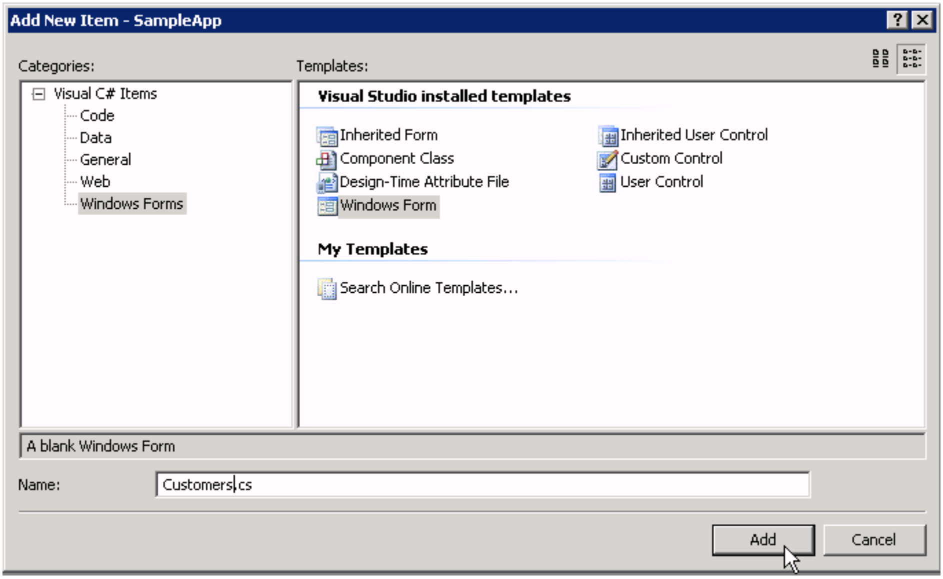
Creating the User Interface for the Windows Mobile Device Application

Goal: Create the user interface for an application that runs on a Windows Mobile device, and accesses the database mobile business objects.

Note: This procedure includes code snippets you can copy and paste for your project. Click *SUP_WM_Custom_Dev_Tutorial_code.zip*to access the text files that include the code snippets for the `CustomerSample`, `CustomerList`, and `CustomerSampleScreen` C# files.

If you are viewing this guide as a PDF, you can locate the referenced document at *http://infocenter.sybase.com/*. Go to *Sybase Unwired Platform 1.5.5 > Tutorial: Windows Mobile Application Development using Custom Development* to launch this PDF.

1. Open the generated Visual Studio solution in Visual Studio 2008.
2. In the Solution Explorer, right-click the *SampleApp* project and select **Add > Windows Form**.
3. In the Add New Item dialog, select **Windows Form** from the Categories and Templates, enter `Customers` as the form name, and click **Add**.


Add New Item - SampleApp

Categories:
- Visual C# Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms

Templates:

Visual Studio installed templates
- Inherited Form
- Component Class
- Design-Time Attribute File
- Windows Form
- Inherited User Control
- Custom Control
- User Control

My Templates
- Search Online Templates...

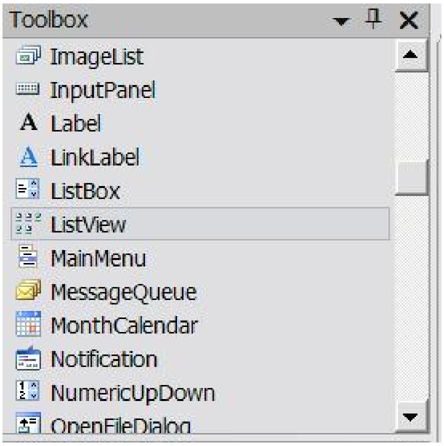
A blank Windows Form

Name: Customers.cs

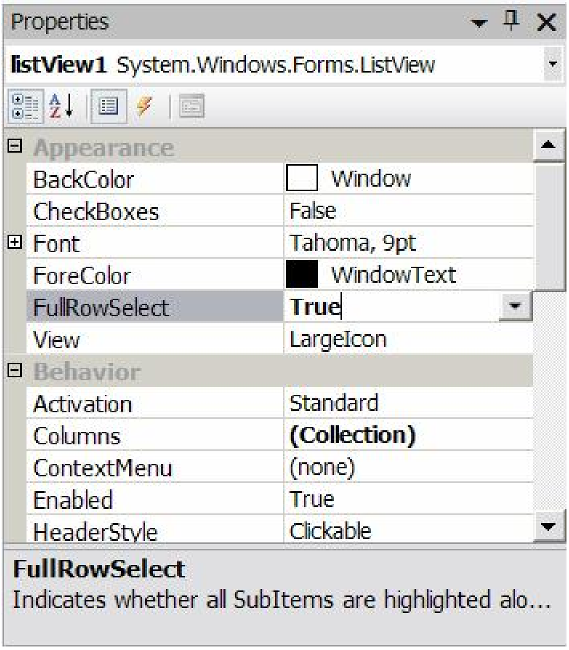
An empty form, **Customers**, displays on the **Customer.cs [Design]** tab.

4. Repeat Step 2 to add another Windows Form to the SampleApp project, and name this one `Customer Details`.

5. Click the **Customers.cs [Design]** tab to go back to the Customers form.
6. From the Toolbox, drag and drop four buttons from **SUP Device Controls** onto the form.
7. Select each button, and in the Properties view, change the **Text** of the buttons to:
   - button1 – `Subscribe`
   - button2 – `Refresh`
   - button3 – `Update`
   - button4 – `Send`
8. In the Toolbox, select **ListView** from **Common Device Controls**, and drag and drop it onto the Customers form.



9. In the Toolbox, select **Textbox** from **Common Device Controls**, and drag and drop it onto the Customers form.
10. In the **Customers** form, click the **ListView**, then in the Properties pane, set FullRowSelect to **True**.



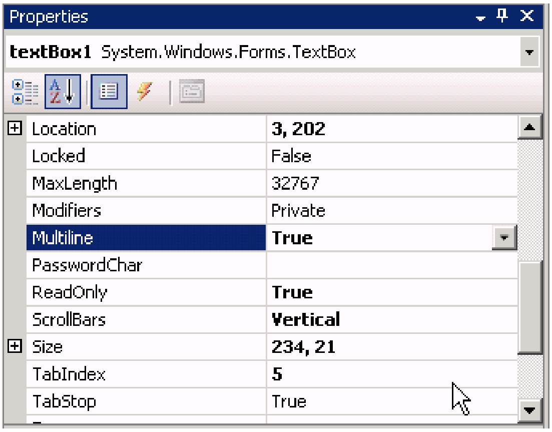
listView1 System.Windows.Forms.ListView

⊟ **Appearance**	
BackColor	□ Window
CheckBoxes	False
⊞ Font	Tahoma, 9pt
ForeColor	■ WindowText
FullRowSelect	**True**
View	LargeIcon
⊟ **Behavior**	
Activation	Standard
Columns	**(Collection)**
ContextMenu	(none)
Enabled	True
HeaderStyle	Clickable

FullRowSelect
Indicates whether all SubItems are highlighted alo…

11. In **Customers**, select the **Textbox**, then in the Properties pane, set these properties:

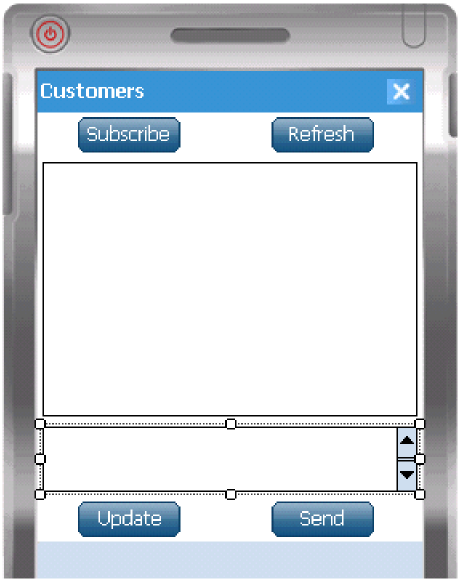
- Scrollbars – Vertical
- Read-only – True
- Multi-line – True



textBox1 System.Windows.Forms.TextBox

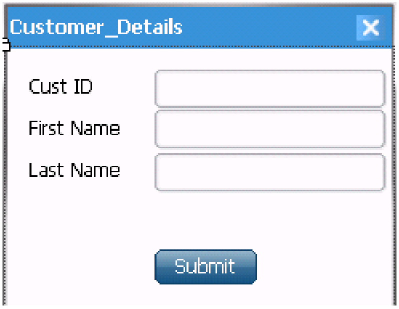
⊞ Location	**3, 202**
Locked	False
MaxLength	32767
Modifiers	Private
Multiline	**True**
PasswordChar	
ReadOnly	**True**
ScrollBars	**Vertical**
⊞ Size	**234, 21**
TabIndex	**5**
TabStop	True

12. Arrange the controls on the Customers form so they look like this:



13. Save the `Customers.cs` form.
14. Click the **Customer Details.cs [Design]** tab.
15. From the Toolbox, drag and drop three labels onto the Customer Details form. Align them on the left side of the form. In the Properties view, in the Text field, rename the labels **Cust ID**, **First Name**, and **Last Name**.
16. From the Toolbox, drag and drop three text boxes onto the Customer Details form and align them to the right of each of the three labels.
17. Select the textbox next to Cust ID and, in the Properties view, change the Read-only property to **True**.
18. From the Toolbox, drag and drop a button from **SUP Device Controls** onto the Customer Details form below the labels and text boxes, and in the Properties view, in the Text field, rename the button to **Submit**.

    The form will look like this:



19. In the Solution Explorer, right-click `Program.cs`, and select **View Code**.

20. Add the following code:

```
/// <summary>
/// The main entry point for the application.
/// </summary>private static Customers _form1 = new Form1();
using System;

using SampleApp;
using System.Windows.Forms;

namespace SampleApp
{
    static class Program
    {
        private static Customers _form1 = new Customers();
        private static Customer_Details _form2 = new 
Customer_Details();
        private static string _custid;

        public static string getCustomer()
        {
            return _custid;
        }

        public static void setCustomer(string custid)
        {
            _custid = custid;
        }

        public static Customers getForm1()
        {
            return _form1;
        }

        public static Customer_Details getForm2()
        {
            return _form2;
        }

        static void Main(string[] args)
        {
            Application.Run(_form1);
        }
    }
}
```

21. In Solution Explorer, in the SampleApp project, right-click **Customers.cs** and select **View Code**.

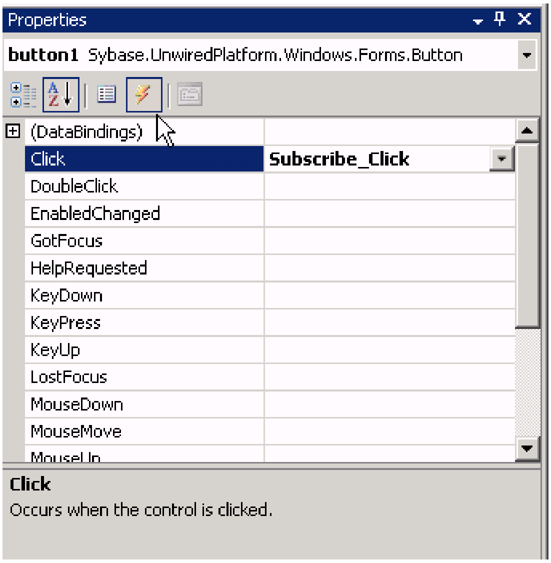
22. Add the following code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SUP101;

namespace SampleApp
{
    public delegate void DelegateAddString(String s);
    public partial class Customers : Form
    {
        DelegateAddString m_DelegateAddString;
        public Customers()
        {
            InitializeComponent();
            m_DelegateAddString = new 
DelegateAddString(this.AddString);
        }

        private void AddString(String s)
        {
            textBox1.Text += s + "\r\n";
            textBox1.SelectionStart = textBox1.Text.Length;
            textBox1.ScrollToCaret();
            textBox1.Refresh();
        }

        private void Subscribe_Click(object sender, EventArgs e)
        {
            //This checkes to see if the database exists, then 
deletes it if it does.
            if (SUP101DB.DatabaseExists())

            {
                SUP101DB.DeleteDatabase();
                AddString("Database deleted");
            }
            else
            {
                //This creates the database.
                SUP101DB.CreateDatabase();

            }
            //To receive callbacks, you must register a 
CallBackHandler
           //with the generated database class, the entity class, 
or both. This
            //code creates and registers a handler to receive 
callbacks.
            MyCallbackHandler callback = new MyCallbackHandler();
            MyCallbackHandler.textBox1 = textBox1;
            SUP101DB.RegisterCallbackHandler(callback);
           //This operation starts background synchronization for 
the database class.
            SUP101DB.StartBackgroundSynchronization();
            AddString("Background synchronization started");
            try
            {
             //This calls SUP101, the package database, user name
and password to connect to the Unwired Server.
                SUP101DB.LoginToSync("supAdmin", "s3pAdmin");
            }
            catch (Exception ex)
            {
                List<LogRecordImpl> lrs =
LogRecordImpl.FindWithQuery(new Sybase.Persistence.Query());
                AddString("log record count=" + lrs.Count);
                for (int i = 0; i < lrs.Count; i++)
                {
                    AddString(lrs[i].Message);
                }
            }
            //For message-based replication, before you can
synchronize MBO changes with the server, you must subscribe to
the
           //mobile application package deployed on the server by
calling SampleAppDB.subscribe(). This also downloads certain data
           //to devices for those that have default values. otify
the server of your subscription to a specific package.
            //If you are not subscribed to the SUP101 package,a
message will tell you to subscribe first.
            SUP101DB.Subscribe();
            AddString("Subscribe request sent");
            AddListView();
        }

        private void refresh_Click(object sender, EventArgs e)
        {
            if (SUP101DB.IsSubscribed())
            {
                Cursor.Current = Cursors.WaitCursor;
                AddString("Refresh data");
                AddDataToListView();
                Cursor.Current = Cursors.Default;
            }
            else
            {
                AddString("Please Subscribe first");
            }
        }

        private void AddListView()
        {
            this.listView1.Clear();
            listView1.Columns.Add("Id", listView1.Width / 4,
HorizontalAlignment.Left);
           listView1.Columns.Add("First Name", listView1.Width /
3, HorizontalAlignment.Center);
          listView1.Columns.Add("Last Name", listView1.Width / 3,
HorizontalAlignment.Right);
            listView1.View = View.Details;
            listView1.FullRowSelect = true;
        }
        private void AddDataToListView()
        {
            this.listView1.Clear();
            listView1.Columns.Add("Id", listView1.Width / 4, 
HorizontalAlignment.Left);
           listView1.Columns.Add("First Name", listView1.Width / 
3, HorizontalAlignment.Center);
          listView1.Columns.Add("Last Name", listView1.Width / 3, 
HorizontalAlignment.Right);
            listView1.View = View.Details;
            listView1.FullRowSelect = true;

            List<Customer> c = Customer.FindAll();
            if (c.Count > 0)
            {
                for (int i = 0; i < c.Count; i++)
                {
                    ListViewItem item = new 
ListViewItem(c[i].Id.ToString());
                    item.SubItems.Add(c[i].Fname);
                    item.SubItems.Add(c[i].Lname);
                    listView1.Items.Add(item);
                }
            }
        }

        private void update_Click(object sender, EventArgs e)
        {
            if (listView1.FocusedItem != null)
            {
                Program.setCustomer(listView1.FocusedItem.Text);
                Program.getForm2().Visible = true;
                Program.getForm1().Visible = false;
            }
            else
                MessageBox.Show("Please select a row");
        }

        private void Send_Click(object sender, EventArgs e)
        {
            Customer.SubmitPendingOperations();
        }

        public class MyCallbackHandler : 
Sybase.Persistence.DefaultCallbackHandler
        {
            public static TextBox textBox1 = new TextBox();

            private void invokeDelegate(string s)
            {
                Customers f = Program.getForm1();
              f.Invoke(f.m_DelegateAddString, new Object[] { s });
            }
            // Called when login fails.
            public override void OnLoginFailure()
            {
                invokeDelegate("Login failed");
            }
            //Called when a replay request succeeds.
            public override void OnReplaySuccess(object o)
            {
                invokeDelegate("Operation replay successful");
            }
            //Called when a replay request fails.
            public override void OnReplayFailure(object o)
            {
                invokeDelegate("Operation Replay failed");
            }
           //Called when the last import message is successfully 
processed regarding the subscribe request.
            public override void OnImportSuccess()
            {
                invokeDelegate("Import successful");
            }
        }
    }

}
```

23. Click the **Customers.cs [Design]** tab to go back to the Customers form design view.

24. Add the event handlers to the buttons:

a) Click the Subscribe button on the form, and in the Properties view for the button, click the Event icon (lightning bolt), then next to the Click databinding, select **Subscribe_Click** from the list.



Click Subscribe_Click

Click

Occurs when the control is clicked.

a) Repeat this step for each button, selecting these events for each Click databinding:

- Refresh – **refresh_Click**
- Update – **update_Click**
- Send – **Send_Click**

25. In Solution Explorer, in the SampleApp project, right-click **Customer Details.cs** and select **View Code**.

26. Add the following code:

```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using SUP101;

namespace SampleApp
{
    public partial class Customer_Details : Form
    {
        Customer thisCustomer;
        public Customer_Details()
        {
            InitializeComponent();
        }
        private void submit_Click(object sender, EventArgs e)
        {
            thisCustomer.Fname = textBox2.Text;
            thisCustomer.Lname = textBox3.Text;
         //When called, the Save method determines, internally, if
it should
          //insert or update data to the client database.
            thisCustomer.Save();
            Program.getForm1().Visible = true;
            Program.getForm2().Visible = false;
        }

        private void AddDataToForm()
        {
            textBox1.Text = Program.getCustomer();
            int id = Int32.Parse(Program.getCustomer());
          //Retrieves data from the local database.
            thisCustomer = Customer.FindByPrimaryKey(id);
            textBox2.Text = thisCustomer.Fname;
            textBox3.Text = thisCustomer.Lname;
        }

       private void Customer_Details_Load(object sender, EventArgs
e)
        {
            AddDataToForm();
        }
        private void Customer_Details_paint(object sender,
PaintEventArgs e)
        {
            AddDataToForm();
        }
    }
}
```

27. In the **Customer_Details.cs [Design]** view, click the Submit button and in the Properties view for the button, add the **submit_Click** event to the Submit button.

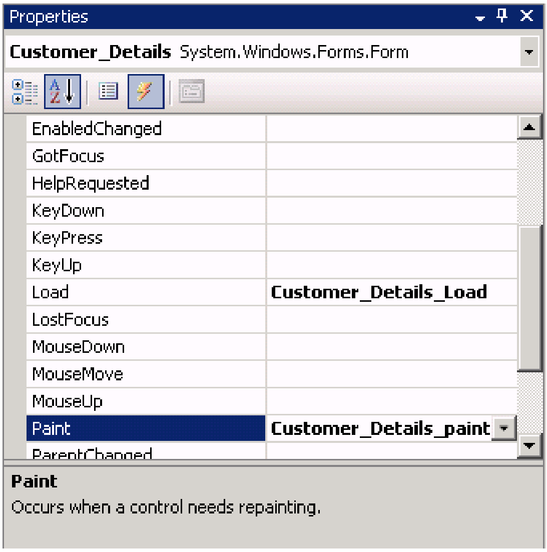
28. Add events to the `Customer Details.cs`:

a) Click the **Customer Details.cs [Design]** tab

b) In the Properties view for Customer_Details.cs, click the Events icon (lightning bolt).

c) In Load, add the **Customer_Details_Load** event.

d) In Paint, add the **Customer_Details_paint** event.



Customer_Details System.Windows.Forms.Form

EnabledChanged	
GotFocus	
HelpRequested	
KeyDown	
KeyPress	
KeyUp	
Load	**Customer_Details_Load**
LostFocus	
MouseDown	
MouseMove	
MouseUp	
Paint	**Customer_Details_paint**
ParentChanged	

Paint
Occurs when a control needs repainting.

29. In the main menu, select **File** > **Save All**.

30. Add the generated C# object API project (from the previous section) into the SampleApp solution. This helps ensure the project includes DLL references needed for compilation.

31. Build the project by pressing `Control+Shift+B`.

Installing Sybase Messaging Runtime

Start the Device Emulator Manager and install Sybase Messaging Runtime.

Prerequisites

Installing Microsoft ActiveSync on page 9

Task

1. In Visual Studio, select **Tools** > **Device Emulator Manager**.
2. From the list of devices, right-click **Windows Mobile 6 Professional Emulator** and select **Connect**.
3. Right-click again on **Windows Mobile 6 Professional Emulator** and select **Cradle**.
   Microsoft ActiveSync appears.
4. If the Microsoft ActiveSync configuration window appears, click **Cancel**.
5. In Microsoft ActiveSync, click **Explore**.
6. In the Mobile Device window, double-click **My Windows Mobile-based Device**.
   The device's file system root folder opens.
7. Navigate to `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\ClientAPI\MoMessaging\wm` and copy the `SUPMessaging_Pro.cab` file to the device's **My Documents** folder.
8. Use File Explorer on the device emulator to browse to the `MyDocuments\SUPMessaging_Pro.cab` file.
9. Click on `SUPMessaging_Pro.cab` once to start the installation.

Registering the Emulator in Sybase Control Center

Goal: Use Sybase Control Center to register the emulator.

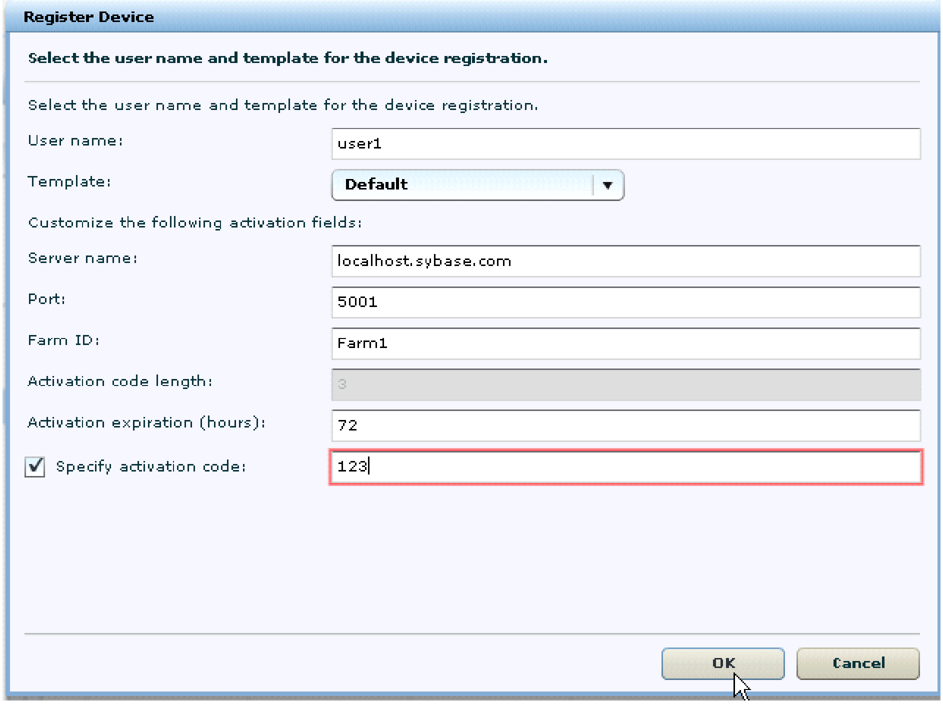
Prerequisites

Complete *Connecting to Sybase Control Center* on page 11.

Task

1. Log in to Sybase Control Center using the `supAdmin/s3pAdmin` user name and password.

2. In Sybase Control Center, select **View** > **Select** > **Unwired Server Cluster Management View**.
3. Select **Device Users**.
4. In the right pane, click **Devices**.
5. Click **Register**.
6. In the Register Device window, enter the required information:
    - User name
    - Server name
    - Port
    - Farm ID – set to 0 (zero).
    - Activation code



Register Device

Select the user name and template for the device registration.

Select the user name and template for the device registration.

User name: user1

Template: Default

Customize the following activation fields:

Server name: localhost.sybase.com

Port: 5001

Farm ID: Farm1

Activation code length: 3

Activation expiration (hours): 72

☑ Specify activation code: 123

Note: The information should match the input on the client and "localhost.sybase.com" should be the actual name of your machine and domain.

7. Click **OK**.

Configuring Connection Settings

Goal: Configure the connection settings on the emulator.

Note: This tutorial used a Windows Mobile 6.0 Professional Edition emulator. If you use a different emulator version, user interface will vary slightly from this tutorial.

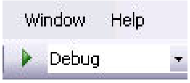
1. On the emulator, select **Start** > **Programs**.
2. Double-click **Sybase Settings**.
3. In the Sybase Settings screen, click **Connection**.
4. In the Connection screen, enter the connection settings. These settings should match the values you used when you registered the device in Sybase Control Center.
    - Server Name – the machine and domain where Unwired Server is running, for example, `localhost.sybase.com`.
    - Server Port – `5001`
    - Farm ID – `Farm1`
    - User name – `user1`
    - Activation Code – `123`

    Note: "localhost.sybase.com" should be the name of the machine where Unwired Server is running.
5. Click **Done**.

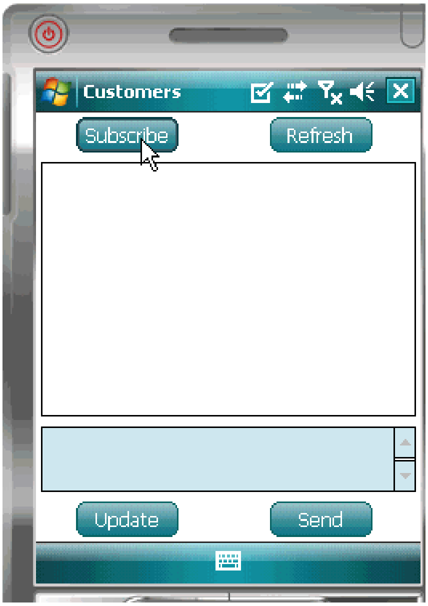
Deploying and Running the Device Application

Goal: Deploy the device application to a Windows Mobile 6 device emulator, and test its functionality.

1. In the Visual Studio Solution Explorer, right-click the **SampleApp** project, and select **Build**.
2. In the Visual Studio toolbar, click the green arrow to the left of **Debug**, select **Windows Mobile 6 Professional Emulator** in the Target Device combobox and click **Deploy**.

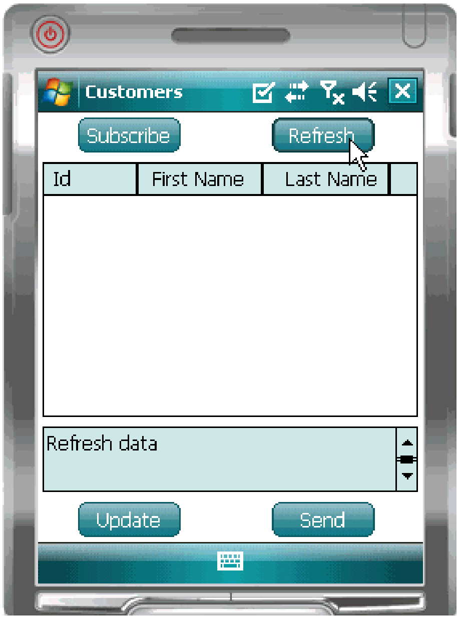
    

    The device application is deployed to the emulator and runs in debug mode.
3. In the device emulator, click **Subscribe**.

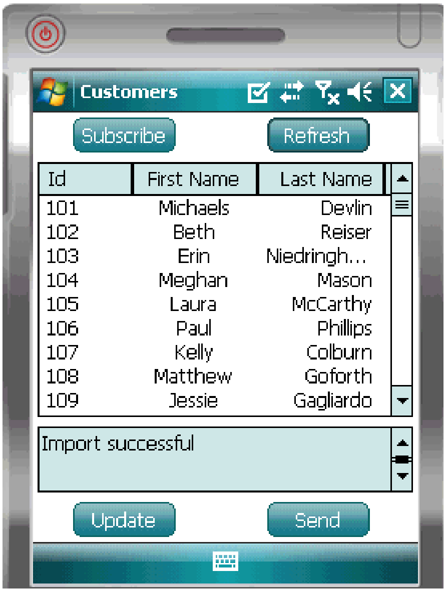


When you are successfully subscribed, a message appears in the text box saying the subscription request was sent.

4. When you see the "Refresh data" message, click **Refresh** to populate the Customers table data in the list view.



Refresh data



Id	First Name	Last Name
101	Michaels	Devlin
102	Beth	Reiser
103	Erin	Niedringh...
104	Meghan	Mason
105	Laura	McCarthy
106	Paul	Phillips
107	Kelly	Colburn
108	Matthew	Goforth
109	Jessie	Gagliardo

Import successful

You see an "Import successful" message. If you see the message but do not see the customer list, click **Refresh** again.

5. Select the customer record, **Beth Reiser**, with the Id 102 from the customer list and click **Update**.
6. In the customer details screen, change the customer's First Name to SUP and click **Submit**.
7. In the Customers screen, click **Refresh**.

   The changed record appears at the bottom of the customer list because the change has not yet been sent to the server.
8. Click **Send**.
9. After you see the "Operation Reply successful" message, click **Refresh**.

   The updated record shows in the customer list.

Learn More about Sybase Unwired Platform

Once you have finished, try some of the other samples or tutorials, or refer to other development documents in the Sybase Unwired Platform documentation set.

Getting Started Tutorials
Try out some of the other getting started tutorials to get a broad view of the development tools available to you.

Advanced Tutorials
Tutorials are available that demonstrate how to use some of Sybase Unwired Platform advanced features.

Check the Sybase Product Documentation Web site regularly for updates: *http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.pubs.docset-SUP-1.5.5/doc/html/title.html*.

Samples
Sample applications are fully developed, working applications that demonstrate the features and capabilities of Sybase Unwired Platform.

Check the Sybase Web site regularly for updates. Navigate to the Sybase Web site, then select *Products > Sybase Unwired Platform > Use tab*: *http://www.sybase.com/products/mobileenterprise/sybaseunwiredplatform?htab=USE*.

Online Help
See the online help that is installed with the product, or the Product Manuals Web site.

Check the Sybase Product Documentation Web site regularly for updates: *http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.pubs.docset-SUP-1.5.5/doc/html/title.html*.

Developer References
See the Developer References to learn about using the API to custom code device applications using the API.

- *Developer Reference for BlackBerry*
- *Developer Reference for iOS*
- *Developer Reference for Mobile Workflow Packages*
- *Developer Reference for Windows and Windows Mobile*

Check the Sybase Product Documentation Web site regularly for updates: *http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.pubs.docset-SUP-1.5.5/doc/html/title.html*.

Javadocs are also available in the installation directory.

Programmer References

See the Programmer References to learn how to use the Administration API and Server API to extend functionality.

- *Reference: Administration APIs* – integrate your own administrative tools with Unwired Platform to monitor and manage Unwired Platform.
- *Reference: Custom Development for Unwired Server* – customize some Unwired Server features.

Check the Sybase Product Documentation Web site regularly for updates: *http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.infocenter.pubs.docset-SUP-1.5.5/doc/html/title.html*.

Javadocs are also available in the installation directory.

Index